COMPSCI 330: Design and Analysis of Algorithms 4/5/2016 and 4/7/2016

Linear Programming Duality and Algorithms

Lecturer: Debmalya Panigrahi *Scribe: Tianqi Song*

1 Overview

In this lecture, we will cover more examples of linear programming and introduce linear programming duality. We will also present several algorithms for solving linear programs.[1]

2 Formulate Problems as Linear Programs

2.1 Maximum Matching

Given a graph $G(V, E)$, find a maximum subset $S \subseteq E$ such that any vertex $v \in V$ is incident to at most one edge in S.

Define

$$X_e = \begin{cases} 1 \ if \ e \ is \ in \ a \ match \\ 0 \ otherwise \end{cases}$$

The integer program is

$$\begin{aligned} &\text{maximize} \sum_{e \in E} X_e \\ &\text{subject to} \sum_{e=(u,v)} X_e \leq 1 \ \forall v \in V \\ &X_e \in \{0, 1\} \end{aligned}$$

We relax our formulation to an LP:

$$\begin{aligned} &\text{maximize} \sum_{e \in E} X_e \\ &\text{subject to} \sum_{e=(u,v)} X_e \leq 1 \ \forall v \in V \\ &X_e \geq 0 \end{aligned}$$

[1] Some materials are from notes by Yilun Zhou, Wenshun Liu and Samuel Haney, and a note by Allen Xiao for COMPSCI 532 in Fall 2015.

2.2 Vertex Cover

Given a graph $G(V, E)$, find a subset $V' \subseteq V$ of minimum size such that $\forall (u, v) \in E, u \in V'$ or $v \in V'$.

Define

$$X_v = \begin{cases} 1 \ \ if \ \ v \in V' \\ 0 \ \ if \ \ v \notin V' \end{cases}$$

The integer program is

$$\begin{aligned} &\text{minimize} \sum_{v \in V} X_v \\ &\text{subject to } X_u + X_v \geq 1 \ \ \forall (u, v) \in E \\ &\qquad\qquad X_v \in \{0, 1\} \ \ \forall v \in V \end{aligned}$$

We relax our formulation to an LP:

$$\begin{aligned} &\text{minimize} \sum_{v \in V} X_v \\ &\text{subject to } X_u + X_v \geq 1 \ \ \forall (u, v) \in E \\ &\qquad\qquad X_v \geq 0 \ \ \forall v \in V \end{aligned}$$

2.3 Minimum Spanning Tree

Given a graph $G(V, E)$ with edge weight w_e, find the minimum spanning tree T.

Define

$$X_e = \begin{cases} 1 \ \ if e \in T \\ 0 \ \ if e \notin T \end{cases}$$

The integer program is

$$\begin{aligned} &\text{minimize} \sum_{e \in E} w_e X_e \\ &\text{subject to} \sum_{e \in (S, \bar{S})} X_e \geq 1 \ \ \forall S \subset V \\ &\qquad\qquad X_e \in \{0, 1\} \ \ \forall e \in E \end{aligned}$$

We relax our formulation to an LP:

$$\begin{aligned} &\text{minimize} \sum_{e \in E} w_e X_e \\ &\text{subject to} \sum_{e \in (S, \bar{S})} X_e \geq 1 \ \ \forall S \subset V \\ &\qquad\qquad X_e \geq 0 \ \ \forall e \in E \end{aligned}$$

2.4 s-t Minimum Cut

Let $G = (V, E)$ be a graph, and let $s, t \in V$. Let edge e have capacity u_e. Our goal is to find a cut $(S, \bar{S})$ such that

$$\sum_{e \in (S, \bar{S})} u_e$$

is minimized, and $s \in S$ and $t \in \bar{S}$. We formulate this problem as an integer program:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} u_e x_e \\ \text{subject to} \quad & \sum_{e \in P} x_e \geq 1 \quad \forall \text{ path } P \text{ from } s \text{ to } t \\ & x_e \in \{0, 1\} \quad \forall e \in E \end{aligned}$$

This is an integer program since values of x_e must be integral. We relax our formulation to an LP.

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} u_e x_e \\ \text{subject to} \quad & \sum_{e \in P} x_e \geq 1 \quad \forall \text{ path } P \text{ from } s \text{ to } t \\ & x_e \geq 0 \qquad \forall e \in E \end{aligned}$$

We call this the fractional minimum s-t cut problem.

3 LP Duality

3.1 Motivating Example

Consider the following LP.

$$\begin{aligned} \text{minimize} \quad & 10x + 10y \\ \text{subject to} \quad & x + 3y \geq 4 \\ & 2x + y \geq 5 \\ & x \geq 0 \\ & y \geq 0 \end{aligned}$$

It is not immediately clear what the solution to this LP is. Instead of trying to find the optimal solution, we try to find a lower bound on the optimal. We multiply each of the first two constraints by some number, and add them.

$$\begin{aligned} & a \cdot (x + 3y \geq 4) \\ + \quad & b \cdot (2x + y \geq 5) \\ \hline \end{aligned} \tag{1}$$

If we set $a = 2$ and $b = 3$, we get

$$\begin{aligned} & 2x + 6y \geq 8 \\ + \quad & 6x + 3y \geq 15 \\ \hline & 8x + 9y \geq 23. \end{aligned}$$

This is significant because the objective is larger than the left side of this constraint. That is,

$$10x + 10y \geq 8x + 9y \geq 23.$$

Therefore, the optimal value of the objective, $10x + 10y$, is at least 23. However, which numbers a and b should we choose to get the best lower bound? Keeping a and b in Equation 1 gives

$$\begin{array}{lllll} & ax & + 3ay & \geq 4a \\ + & 2bx & + by & \geq 5b \\ \hline & (a + 2b)\, x & + (3a + b)\, y & \geq 4a + 5b. \end{array}$$

We want to maximize the right side of this constraint, $4a + 5b$ to get the highest lower bound possible. Additionally, the left side of the constraint should not be more than the objective. We can write this as follows.

$$\begin{array}{rl} \text{maximize} & 4a + 5b \\ \text{subject to} & a + 2b \leq 10 \\ & 3a + b \leq 10 \\ & a \geq 0 \\ & b \geq 0 \end{array}$$

This is just another LP! We call this LP the dual, and the original LP the primal. The dual LP has one constraint for each variable in the primal, and one variable for each constraint in the primal.

3.2 Primal-Dual Pairs

Suppose we have the following minimization program:

$$\begin{array}{rl} \min & c^\intercal x \\ \text{s.t.} & Ax \geq b \\ & x \geq 0 \end{array}$$

Let the optimal solution be x^*. It's straightforward to show that the value of the optimal solution is below some threshold α: demonstrate any feasible x with value α. The optimal must do at least as well.

$$c^\intercal x = \alpha \implies c^\intercal x^* \leq \alpha$$

The opposite direction is not as easy – how would one show that $c^\intercal x^* \geq \alpha$? We cannot use the same trick of finding a solution of value α. Suppose instead we find $y \geq 0$ such that:

$$A^\intercal y \leq c$$

Claim 1. *$b^\intercal y$ is a lower bound on $c^\intercal x^*$.*

Proof. By feasibility:

$$Ax^* \geq b$$

Left multiply by $y^\intercal$.

$$y^\intercal A x^* \geq y^\intercal b$$

Apply our assumption on $y^\intercal A$.

$$c^\intercal x^* \geq y^\intercal b = b^\intercal y$$

□

Clearly, the larger the value of $b^\intercal y$, the better the bound is. The constraints on y form a new linear program:

$$\begin{aligned} \max \quad & b^\intercal y \\ \text{s.t.} \quad & A^\intercal y \leq c \\ & y \geq 0 \end{aligned}$$

We formalize this notion as LP *duality*.

Definition 1. *For a **primal** (P) linear program in the form:*

$$\begin{aligned} \min \quad & c^\intercal x \\ \textit{s.t.} \quad & Ax \geq b \\ & x \geq 0 \end{aligned}$$

*The **dual** (D) linear program is:*

$$\begin{aligned} \max \quad & b^\intercal y \\ \textit{s.t.} \quad & A^\intercal y \leq c \\ & y \geq 0 \end{aligned}$$

If the primal is a canonical form maximization problem, swap P and D above.

Fact 2. *If D is the dual of P, then the dual of D is P. "Dual of the dual is the primal."*

3.3 Weak Duality

The claim we proved earlier, that the maximization dual *lower bounds* the minimization primal, is also known as *weak duality*.

Theorem 3. *Let x be any feasible primal solution, y be any feasible dual solution. The **weak duality** theorem states that, for minimization primal, maximization dual:*

$$c^\intercal x \geq b^\intercal y$$

Alternatively, for maximization primal, minimization dual:

$$c^\intercal x \leq b^\intercal y$$

Proof. Same as the proof for Claim 1. You may have noticed by now that most of our results have mirrored versions for when primal is minimization or maximization. These are more or less exchangeable (we can always switch to the dual problem as our "primal", as suggested by Fact 2). □

Example 1. A different linear program for flow based on flow decomposition.

Let P be the s-t paths in the flow decomposition.

$$\begin{aligned} \max \quad & \sum_{p \in P(s,t)} f(p) \\ \text{s.t.} \quad & \sum_{p:(v,w)\in p} f(p) \le u(v,w) \quad \forall (v,w) \in E \\ & f(p) \ge 0 \end{aligned}$$

The dual is:

$$\begin{aligned} \min \quad & \sum_{(v,w)\in E} u(v,w)\ell(v,w) \\ \text{s.t.} \quad & \sum_{(v,w)\in p} \ell(v,w) \ge 1 \qquad \forall p \in P(s,t) \\ & \ell(v,w) \ge 0 \end{aligned}$$

One can interpret this as: the length function with minimum volume such that $d_\ell(s,t) \ge 1$. For any s-t cut $(S, \overline{S})$, where $d_\ell(s,t)$ is the distance from s to t under ℓ,

$$\ell(v,w) = \begin{cases} 1 & \text{if } (v,w) \in (S,\overline{S}) \\ 0 & \text{otherwise} \end{cases}$$

is feasible, and therefore the max-flow $\le$ min-cut by merely applying weak duality.

Claim 4. *For any feasible fractional $\ell(\cdot)$, there is an integral $\ell'(\cdot)$ such that $\sum u(e)\ell'(e) \le \sum u(e)\ell(e)$. For an optimal $\ell^*(\cdot)$, there is an integral solution of equal value.*

Proof. We will provide a randomized algorithm for rounding any fractional solution $\ell(\cdot)$ to an integral $\ell'(\cdot)$ where:

$$\mathbb{E}\left[\sum_{e\in E} u(e)\ell'(e)\right] \le \sum_{e\in E} u(e)\ell(e)$$

At least one randomized outcome is at least as low as the expectation, proving the claim.

We will use each $\ell'(v,w)$ as a 0-1 random variable. Let $d_\ell(v)$ be the shortest distance of v from t under edge lengths $\ell(\cdot)$. We will assign to each vertex a potential $d(v) := d_\ell(v)$. By the properties of shortest distances:

$$d(v) \le d(w) + \ell(v,w) \qquad \forall (v,w) \in E$$

By the constraints, we must also have $d(s) = d_\ell(s) \ge 1$. To round, we choose $r \in (0,1)$ and let $S = \{v \mid d(v) \ge r\}$. If (v,w) crosses the S cut, we assign $\ell'(v,w) = 1$.

$$\Pr\left[\ell'(v,w) = 1\right] = \frac{d(v) - d(w)}{d(s)} \le \frac{\ell(v,w)}{1}$$

Now in expectation:

$$\begin{aligned}\mathbb{E}\left[\sum_{e\in E} u(e)\ell'(e)\right] &= \sum_{e\in E} u(e)\Pr\left[\ell'(e)=1\right] \\ &\leq \sum_{e\in E} u(e)\ell(e)\end{aligned}$$

As desired. □

Corollary 5. *For primal-dual pair P and D, either of the following must hold.*

1. *P and D are feasible and bounded.*
2. *One of P and D is feasible but unbounded; then the other must be infeasible.*
3. *Both P and D are infeasible.*

4 Linear Programming Algorithms

In this section, we present several algorithms for solving linear programs. We will mostly refer to linear programs in canonical form:

$$\begin{aligned}\min \quad & c^\intercal x \\ \text{s.t.} \quad & Ax \geq b \\ & x \geq 0\end{aligned}$$

Following that, we will introduce the idea of *separation oracles*, on which some of algorithms are based.

4.1 Preliminary

Earlier, we said that the feasible space of linear programs formed a convex polyhedron:

Definition 2. *A **polyhedral set** is the intersection of a finite number of half-spaces in $\mathbb{R}^n$.*

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$$

A linear program minimizes/maximizes some objective over points in P.

Definition 3. *A **vertex** of a polyhedral set P is $x \in P$ where for some $y \in P$:*

$$x - y \in P \text{ and } x + y \in P \implies y = 0$$

In other words, x is not a convex combination of any other points in P.

As the name would suggest, vertices occur on the geometric "corners" of the polyhedron. Intuitively, our definition says that it's impossible to move forward *and* backward in *any* direction from x. We now tie this concept back to linear programs:

Definition 4. *A **basic feasible solution** $x \in P$ is one where n linearly independent constraints are tight.*

Lemma 6. *Vertices and basic feasible solutions are equivalent.*

Proof. We prove both directions.

1. (Vertex $\implies$ BFS)

   By contrapositive. Suppose $x \in P$ is not a basic feasible solution. Let each a_i' be a tight constraint:

   $$a_i' x = b_i$$

   And let the submatrix of tight constraints be A'. Since x is not a basic feasible solution, there are at most $n-1$ linearly indpendent constraints in A'. Since A' is not full rank, its null space $\text{null}(A')$ has dimension at least 1. For any $y \in \text{null}(A')$, its projection onto A' is 0.

   $$\text{proj}_{A'}(y) = 0$$

   Subsequently:

   $$\begin{aligned} A'(x+y) = A'x &= b \\ A'(x-y) = A'x &= b \end{aligned}$$

   Now consider some other constraint in this null space (not tight) a_i. If the problem is not under-constrained, there is at least one.

   $$a_i x > b_i$$

   We can therefore choose a $y \in \text{null}(A'), y \neq 0$ where:

   $$\begin{cases} a_i(x+y) = b_i & \text{if } a_i y > 0 \\ a_i(x-y) = b_i & \text{if } a_i y < 0 \end{cases}$$

   This is the direction y which fails the vertex definition, so x is not a vertex.

2. (Vertex $\impliedby$ BFS)

   Suppose that $x \in P$ is not a vertex. Then there exists some line in direction y such that $x+y, x-y$ are both in P. For every tight constraint i (by definition):

   $$\begin{aligned} a_i x &= b_i \\ a_i(x+y) &\leq b_i \\ a_i(x-y) &\leq b_i \end{aligned}$$

   Therefore $a_i y = 0$. However, the a_i form a rank n matrix by definition of basic feasible solution, and therefore it must be that $y = 0$.

□

Lemma 7. *Any bounded LP in standard form has an optimum at a basic feasible solution.*

Proof. Let x be optimal for P. If x is not a basic feasible solution, there are less than n linearly independent tight constraints. We will move in a direction which increase the number of tight constraints, without decreasing the objective value. As before, let A' be the submatrix of tight constraints. We can find $y \in \text{null}(A')$ where:

$$A'y = 0$$

For sufficiently small $\epsilon > 0$:

$$x \pm \epsilon y$$

is also feasible. Moreover:

$$c^\intercal(x \pm \epsilon y) = c^\intercal x \pm \epsilon c^\intercal y$$

Optimality of x means that

$$c^\intercal y = 0$$

The feasible space is bounded, so one of these directions ($\pm y$) will be bounded. We move x in that direction until a constraint becomes tight. □

4.2 Simplex Algorithm (Dantzig 1947)

Simplex is a class of algorithms which solve linear programs by moving from vertex to vertex until an optimal solution is reached. The variables which require tight dual constraints are called *basic* variables (alternatively, *non-basic*). Simplex swaps a basic variable for a non-basic variable in an operation known as a *pivot*. Gaussian elimination can be used to find the new vertex.

1. (*Phase 1*)

   Find an initial basic feasible solution x, with tight constraints $T \subseteq \{a_1, \ldots, a_m\}$. This can be done by solving another LP which has $x = 0$ as an initial BFS.

2. (*Phase 2*)

   Repeatedly perform cost-improving pivots (swapping out elements of T) until x is optimal (no pivot improves the cost). Since the size of T does not change, x is always a basic feasible solution.

3. Output optimal x.

The specific algorithm depends on the *pivoting rule*, which describes the vertex to be explored next. There is no known pivoting rule for which the simplex algorithm is worst-case sub-exponential time, it is quite successful in practice. For more information on hard instances for simplex-style algorithms, look up the Klee-Minty cube.

4.3 Ellipsoid Algorithm (Khachiyan 1980)

The ellipsoid algorithm is a weakly polynomial algorithm which solves for LP feasiblity, due to Khachiyan. However, LP optimality and LP feasibility are equivalent, so this still "solves" the linear program.

The basic idea for the ellipsoid algorithm is:

1. Maintain an ellipsoid containing the polyhedron P.

2. Check if the center of the ellipsoid is inside P. If so, done.
3. If not, find a separating hyperplane, parallel to the violated constraint and through the ellipsoid center, and split the ellipsoid in half.
4. Enclose the half-ellipsoid containing P in a minimal ellipsoid containing it. Recurse on this smaller ellipsoid.

4.3.1 Separation Oracles

Something to notice is that the ellipsoid algorithm does not require the linear program to have a polynomial number of constraints (it need not look at the linear program at all). It only requires a polynomial time *separation oracle*.

Definition 5. *A **separation oracle** reports whether a point is feasible, or else gives a violated constraint. Formally, given candidate solution x for $P = \{Ax \geq b\}$, show:*

1. *$x \in P$ or*
2. *$x \notin P$ and constraint a_i where:*

$$a_i x < b_i$$

If the number of constraints *is* polynomial, we could just check each one manually.

Example 2. Recall the path-based LP for maximum flow:

$$\begin{aligned} \max \quad & \sum_{p \in P(s,t)} f(p) \\ \text{s.t.} \quad & \sum_{p:(v,w)\in p} f(p) \leq u(v,w) \quad \forall (v,w) \in E \\ & f(p) \geq 0 \end{aligned} \qquad \begin{aligned} \min \quad & \sum_{(v,w)\in E} u(v,w)\ell(v,w) \\ \text{s.t.} \quad & \sum_{(v,w)\in p} \ell(v,w) \geq 1 \qquad \forall p \in P(s,t) \\ & \ell(v,w) \geq 0 \end{aligned}$$

The primal problem has an exponential number of variables, but a polynomial number of constraints (vice versa for the dual). An exponential number of dual constraints seems like a problem to verify, but we have a separation oracle.

A polynomial time separation oracle for the dual is the *shortest path* under $\ell(\cdot)$. If the length of the shortest path is < 1, then the shortest path is a violated dual constraint. Otherwise, when the length of the shortest path is ≥ 1, we know that *all* paths are at least length 1 under $\ell(cdot)$, and therefore the length function is feasible. Such a shortest path can be computed in polynomial time, using Djikstras or Bellman-Ford.

4.4 Interior Point Algorithms

Interior point algorithms stay inside the feasible space, and gradually approach the polytope boundary. These algorithms use a potential function to measure the duality gap and distance from the boundary, and a barrier function which makes the boundary unattractive. When the algorithm approaches the boundary, it maps the polyhedron to a new space where the boundary is farther away. Contrast to the simplex algorithm, which traces along the surface of the polytope. These are more practical than ellipsoid.